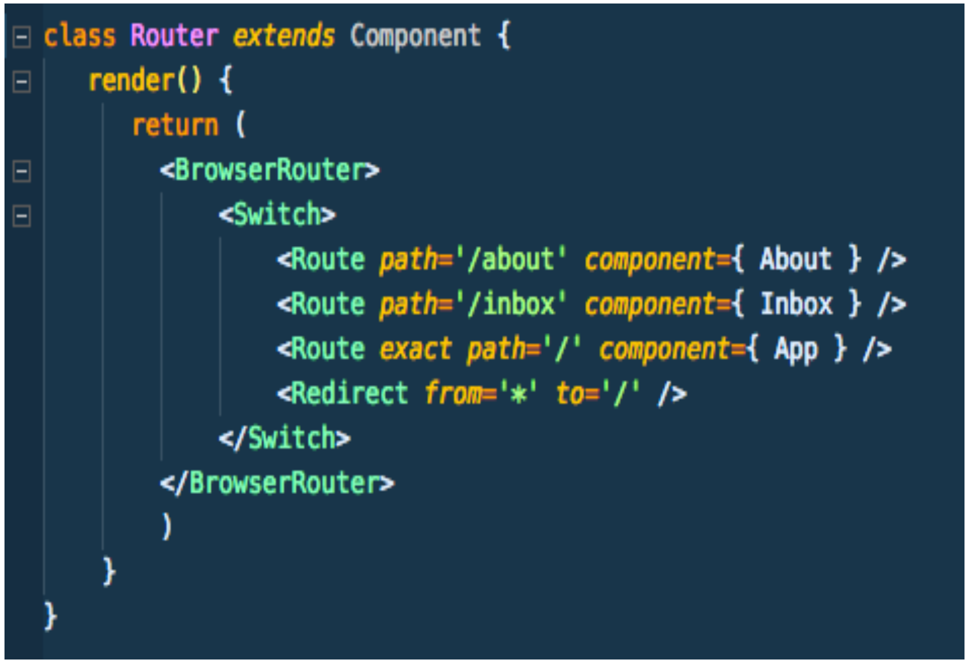
The React Router

Introduction

- **A separate library.**
- **Allows multiple views and flows in an app.**
- **Keeps the URL in sync with what's being displayed.**
- **Supports traditional web principles:**
  1. **Addressability**
  2. **Information sharing.**
  3. **Deep linking.**
  - **1st generation AJAX apps violated these principles**

Basic routing configuration

	URL	Components
1	/	App (Home)
2	/about	About
3	/inbox	Inbox

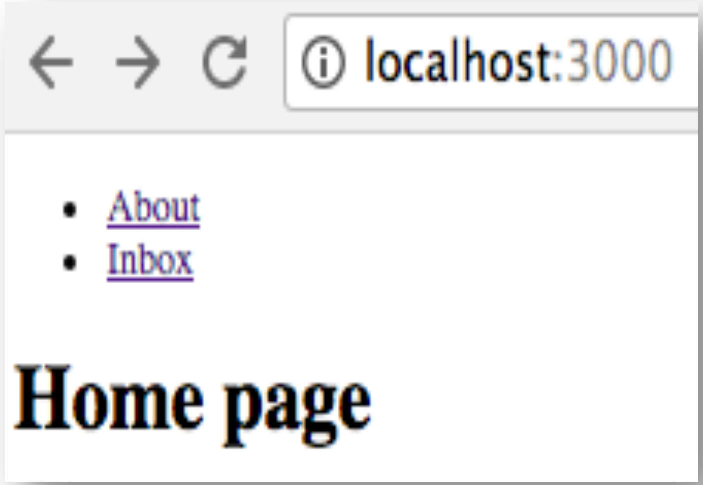


```
class Router extends Component {
  render() {
    return (
      <BrowserRouter>
        <Switch>
          <Route path='/about' component={ About } />
          <Route path='/inbox' component={ Inbox } />
          <Route exact path='/' component={ App } />
          <Redirect from='*' to='/' />
        </Switch>
      </BrowserRouter>
    )
  }
}
```

- **Ref.** src/sample1/
- <Switch> - **like conventional case-switch statement.**
  - **Matches browser URL to nested** Route **components'** path **prop value.**
  - **Matching supports regular expression pattern matching.**
  - **Use** exact **argument for precision.**
  - **Use** <Redirect> - to **avoid 404-type error.**
- ReactDOM.render() **passed an app's Router component**.

Hyperlinks

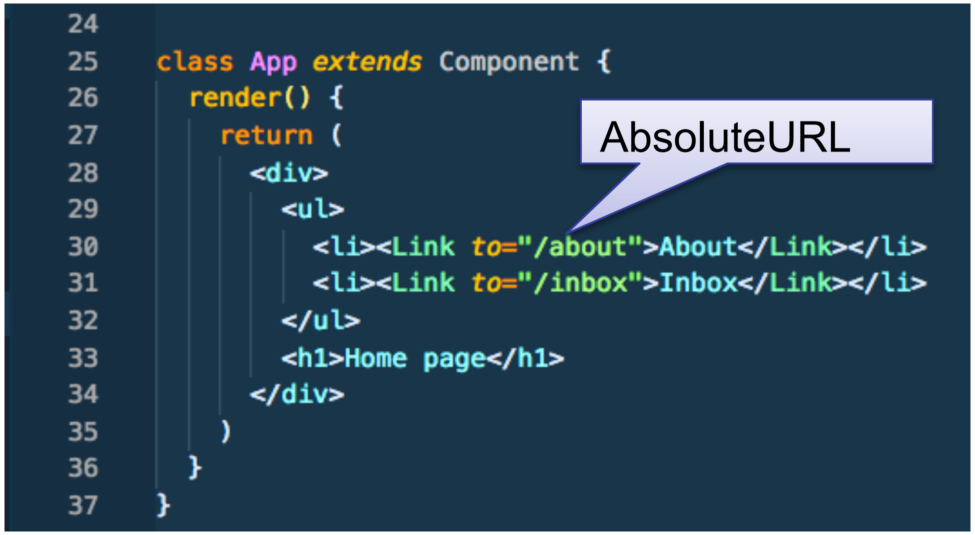
- **Use the** <Link> **component for internal links.**
  - **Use anchor tag for external links -** <a href . . . . . >
- **EX. Ref**. src/sample2/)



localhost:3000

- About
- Inbox

Home page



```
class App extends Component {
  render() {
    return (
      <div>
        <ul>
          <li><Link to="/about">About</Link></li>
          <li><Link to="/inbox">Inbox</Link></li>
        </ul>
        <h1>Home page</h1>
      </div>
    )
  }
}
```

- <Link> **gives access to other useful router properties.**
- **Use** <LinlContainer> **when link wraps other 3rd party component, e.g. Bootstrap-React <Buttom />**

Dynamic segments.

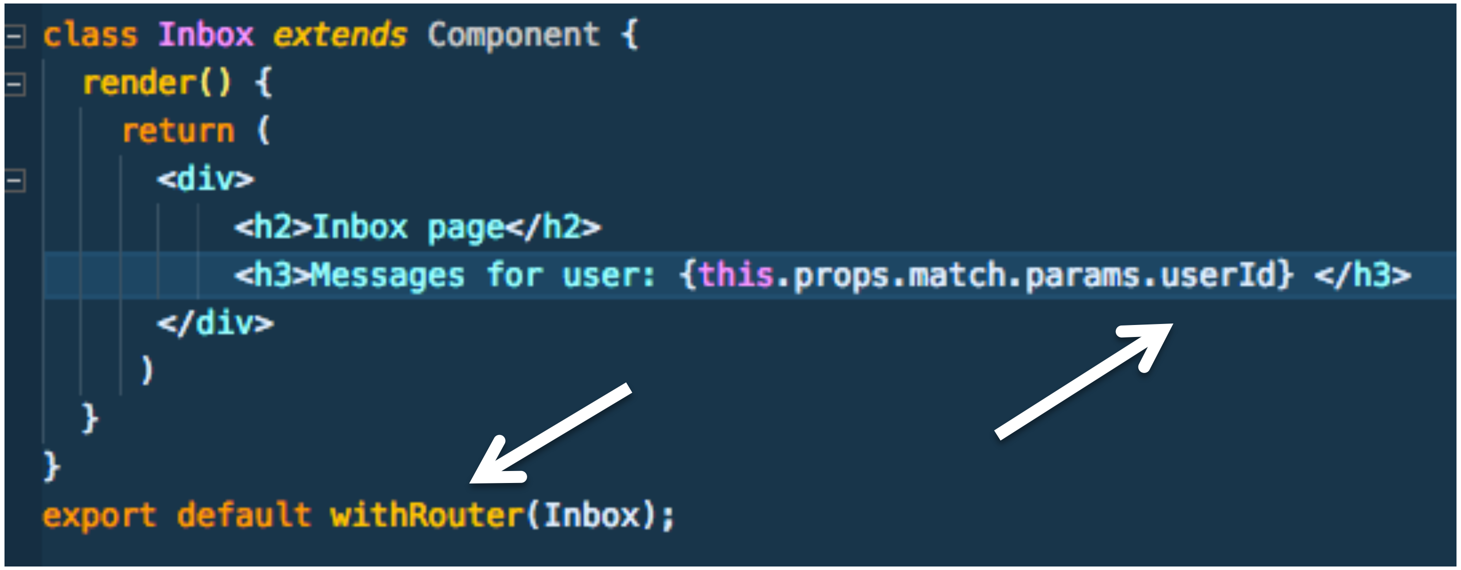
- Parameterized URLs**, e.g. /users/22, /users/12/purchases**
  - **How do we declare a parameterized path?**
  - **How does a component access the parameter value?**

- **Ex: Ref** src/sample3/**.**
  - **Suppose the** Inbox **component must show messages for a user specified in the browser URL e.g /inbox/123**

```
……
<Route path='/inbox/:userId' component={ Inbox } />
……..
```

The colon (:) indicates a parameter position. Parameter name (e.g. userId) is arbitrary.

Dynamic segments.



```
class Inbox extends Component {
  render() {
    return (
      <div>
        <h2>Inbox page</h2>
        <h3>Messages for user: {this.props.match.params.userId} </h3>
      </div>
    )
  }
}
export default withRouter(Inbox);
```

- withRouter() **function - makes routing-related information available to a component by** injecting it **with extra props.**
  - props.match.params.(parameter-name)
  - props.history

Nested Routes

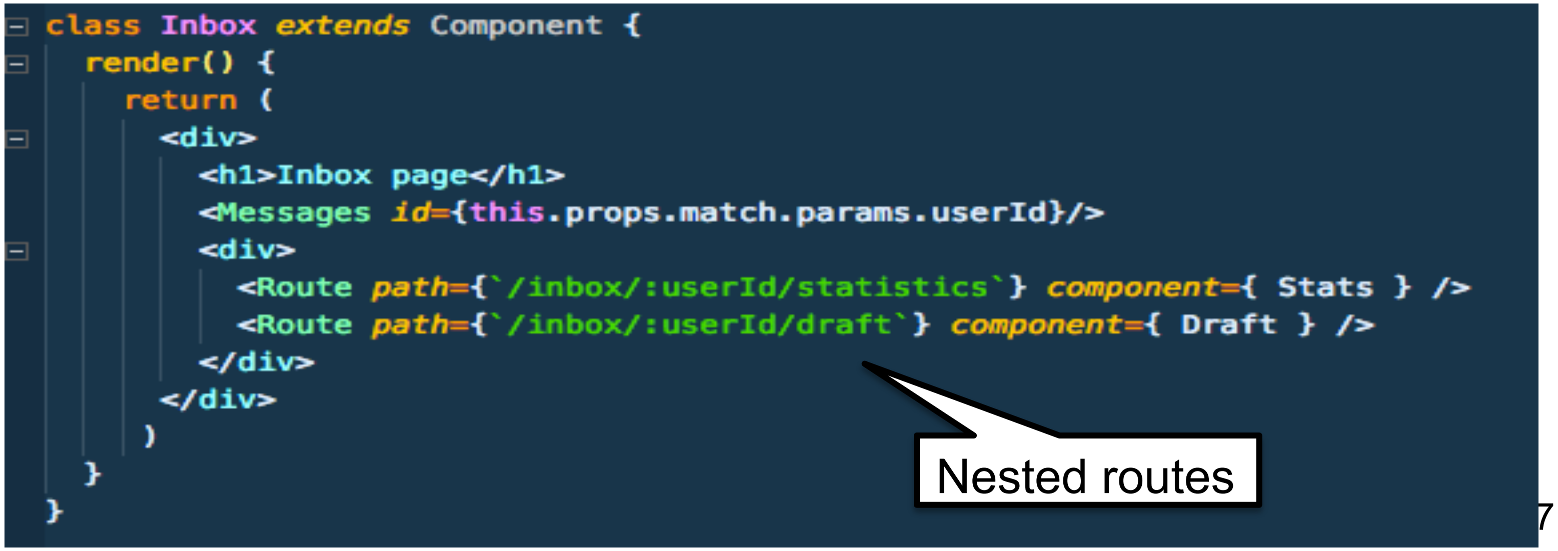
- **EX.: See** src/sample4/**.**

  Objective: Given the route:

  <Route path='/inbox/:userId' component={ Inbox } />,

  when the browser URL is:

  1. /inbox/XXX/statistics **then render Inbox +** Stats **components.**
  2. /inbox/XXX/draft **then render Inbox +** Drafts **components.**



```
class Inbox extends Component {
  render() {
    return (
      <div>
        <h1>Inbox page</h1>
        <Messages id={this.props.match.params.userId}/>
        <div>
          <Route path={`/inbox/:userId/statistics`} component={ Stats } />
          <Route path={`/inbox/:userId/draft`} component={ Draft } />
        </div>
      </div>
    )
  }
}
```

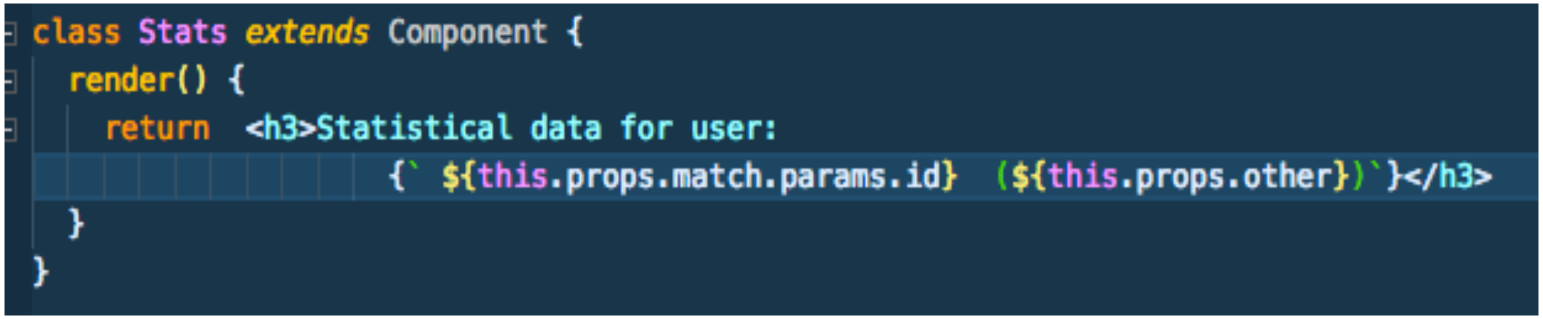
Alternative <Route> API feature.

- **To-date: <Route path={…URL path…}** component**={ ComponentX}>**
- **Disadv.: We cannot pass custom props to the component.**
- **Alternative:**

  <Route path={…URL path…} remder**={…function….}>**
  - **render function must return a component.**

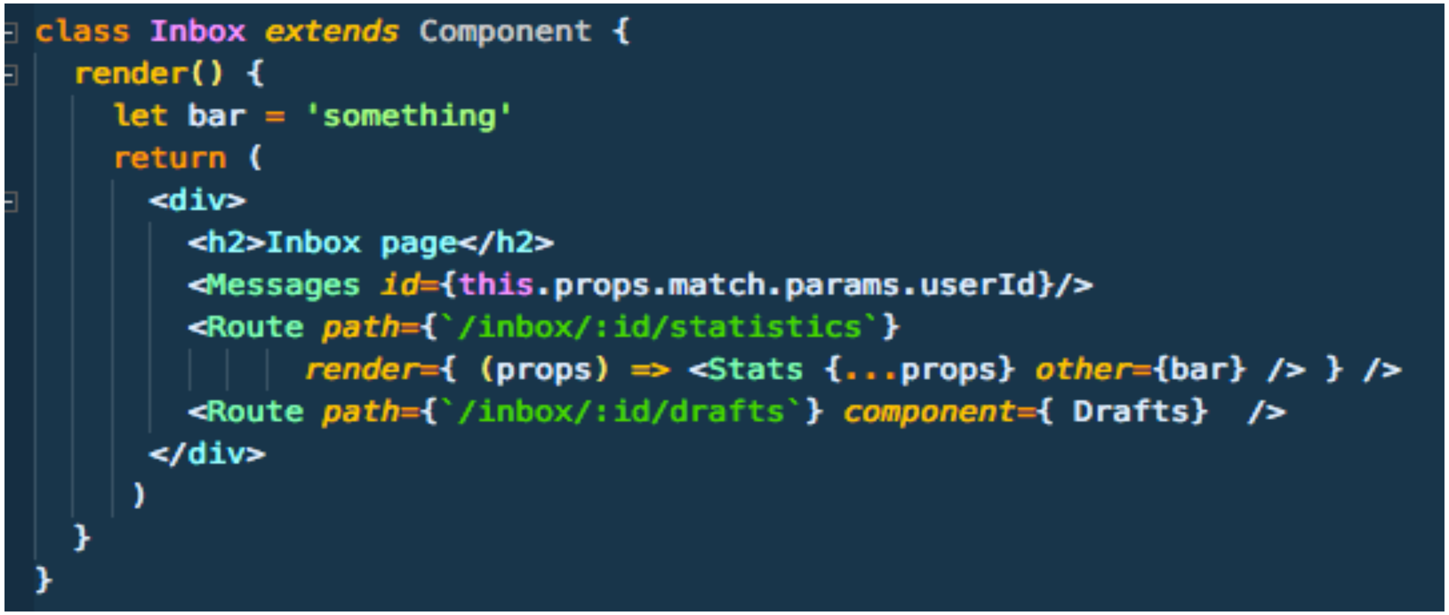
- **EX.: See** /src/sample5/**.**

  Objective: Pass a custom prop to the <Stats> **component.**



```
class Stats extends Component {
  render() {
    return <h3>Statistical data for user:
                  {` ${this.props.match.params.id}  (${this.props.other})`}</h3>
  }
}
```

Alternative <Route> API feature.



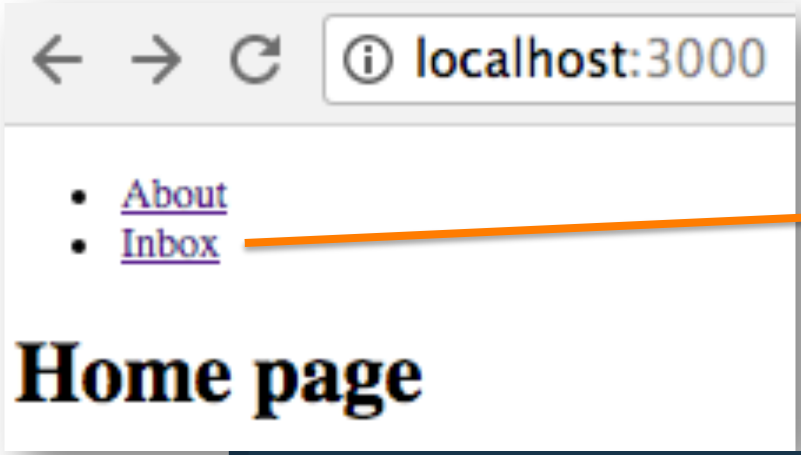
```
class Inbox extends Component {
  render() {
    let bar = 'something'
    return (
      <div>
        <h2>Inbox page</h2>
        <Messages id={this.props.match.params.userId}/>
        <Route path={`/inbox/:id/statistics`}
              render={ (props) => <Stats {...props} other={bar} /> } />
        <Route path={`/inbox/:id/drafts`} component={ Drafts}  />
      </div>
    )
  }
}
```

```
render= (props) => {
      . . . Some logic . . .. . .
      return <ComponentX prop1=…  prop2=….     />
}
```

Where props **is the standard props object passed by <Route> to its subordinate, e.g. the arrow function.**

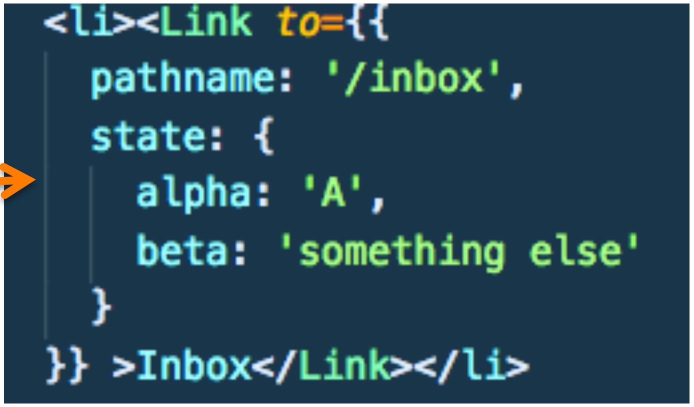
Extended <Link> API

- **Objective: Passing additional props via a <Link>.**
- **EX.: See** /src/sample6/**.**

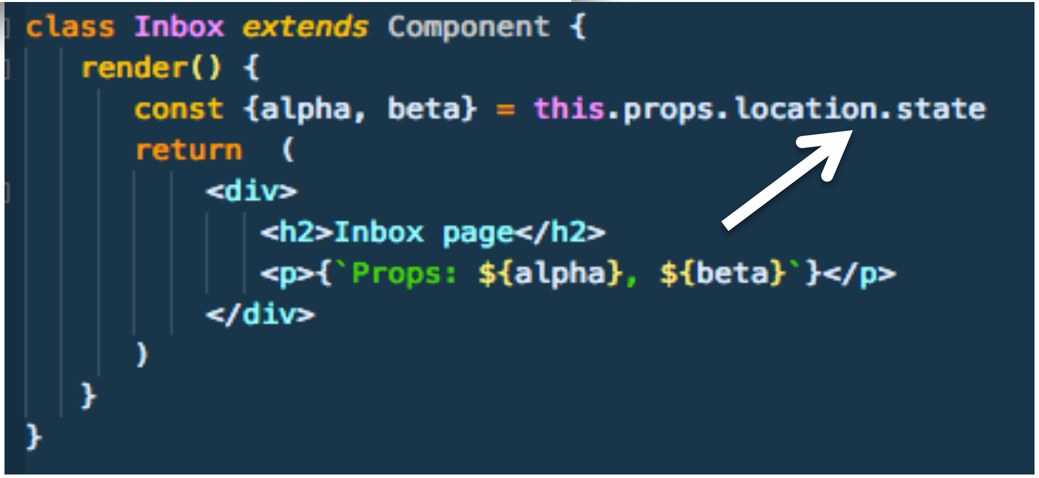


- About
- Inbox

Home page



```
<li><Link to={{
  pathname: '/inbox',
  state: {
    alpha: 'A',
    beta: 'something else'
  }
}} >Inbox</Link></li>
```

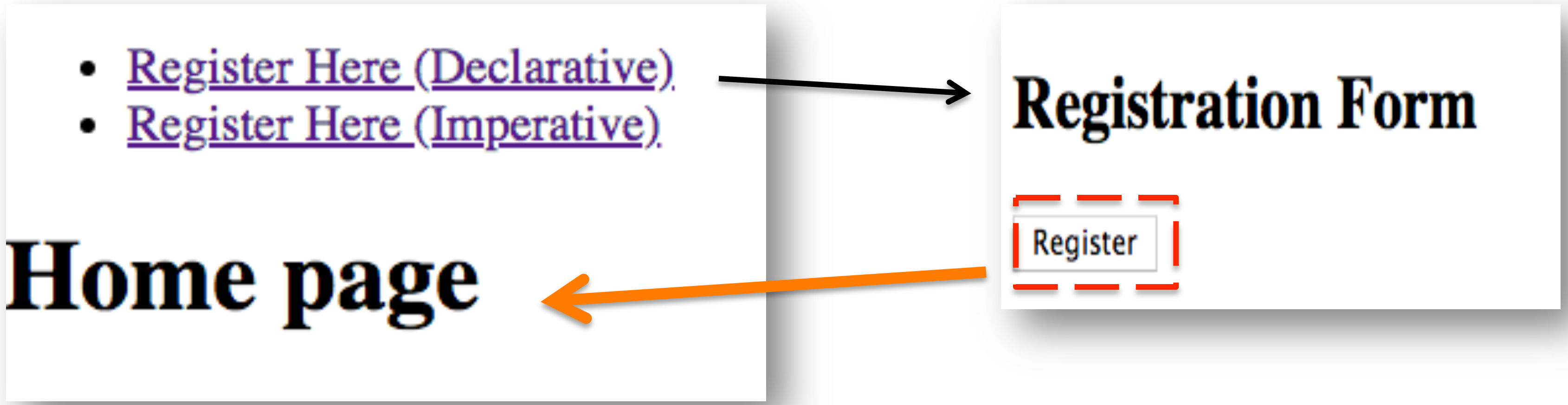


```
class Inbox extends Component {
  render() {
    const {alpha, beta} = this.props.location.state
    return  (
      <div>
        <h2>Inbox page</h2>
        <p>{`Props: ${alpha}, ${beta}`}</p>
      </div>
    )
  }
}
```

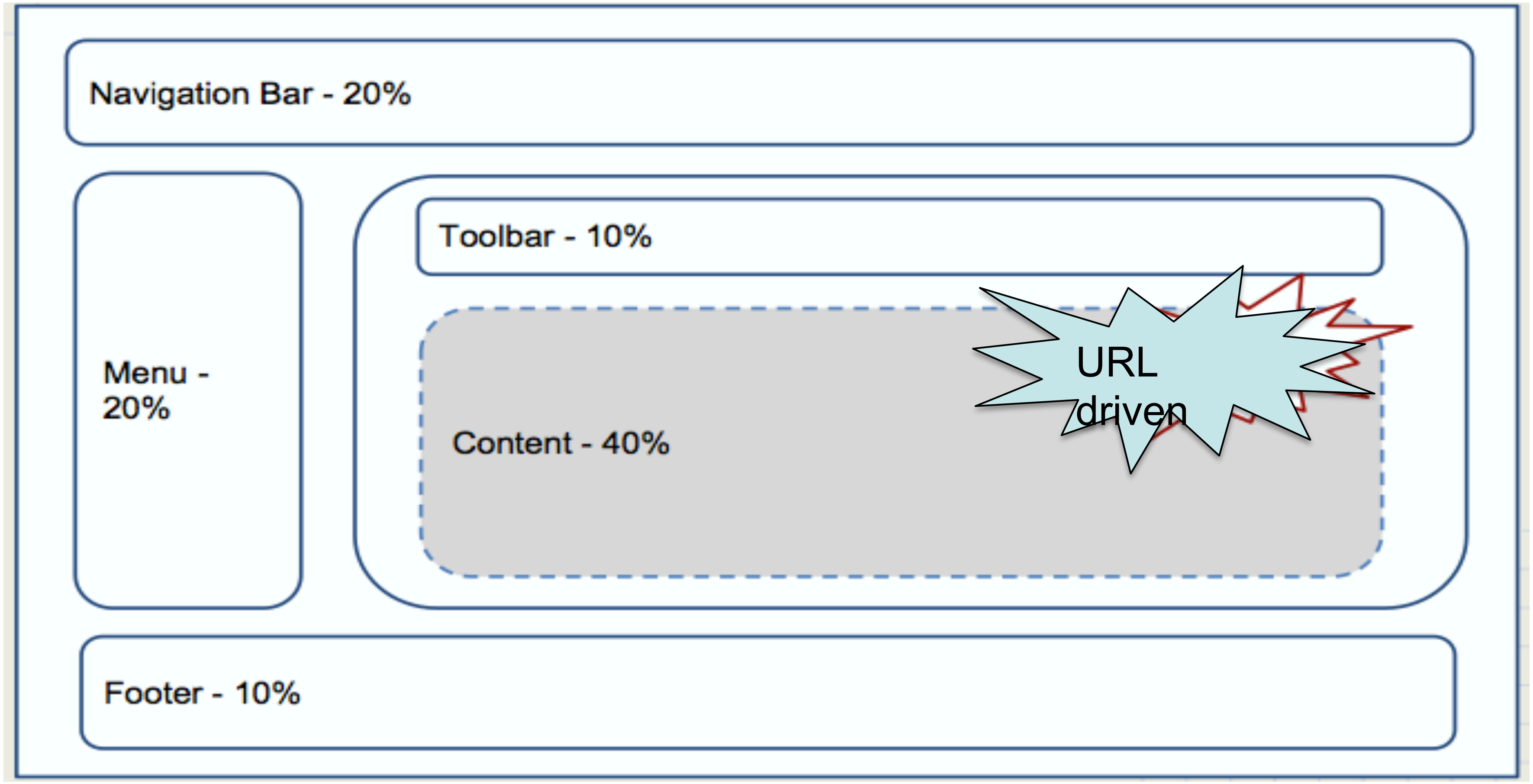
Programmatic Navigation.

- **Performing navigation in JavaScript.**
- **Two options:**
  1. **Declarative – requires state; uses <Redirect>.**
  2. **Imperative – requires withRouter() ; uses this,props.history**

- **EX.: See** /src/sample7/**.**

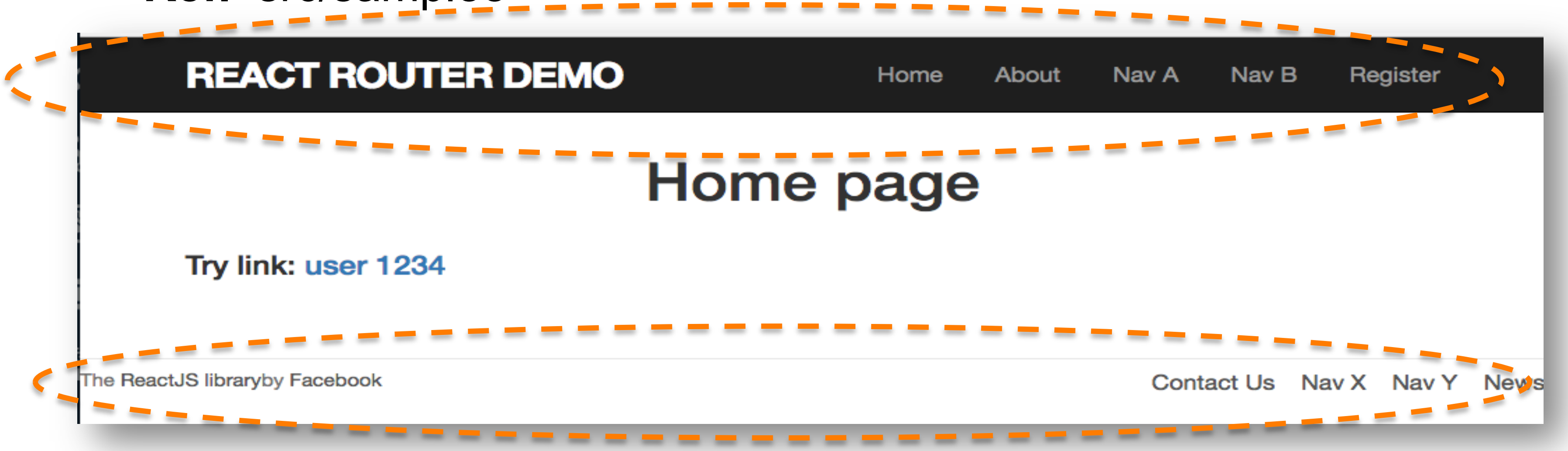


Typical Web app layout



Persistent elements/components

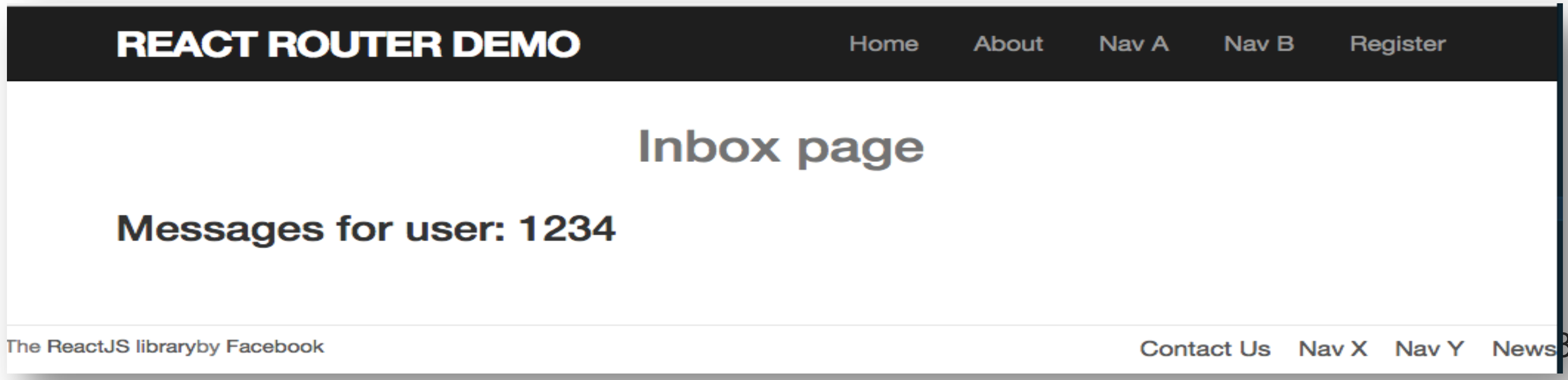
- **Use cases: Headers. Footers. Side menus**
- **Ref.** src/sample8


REACT ROUTER DEMO Home About Nav A Nav B Register

Home page

Try link: user 1234

The ReactJS libraryby Facebook Contact Us Nav X Nav Y News


REACT ROUTER DEMO Home About Nav A Nav B Register

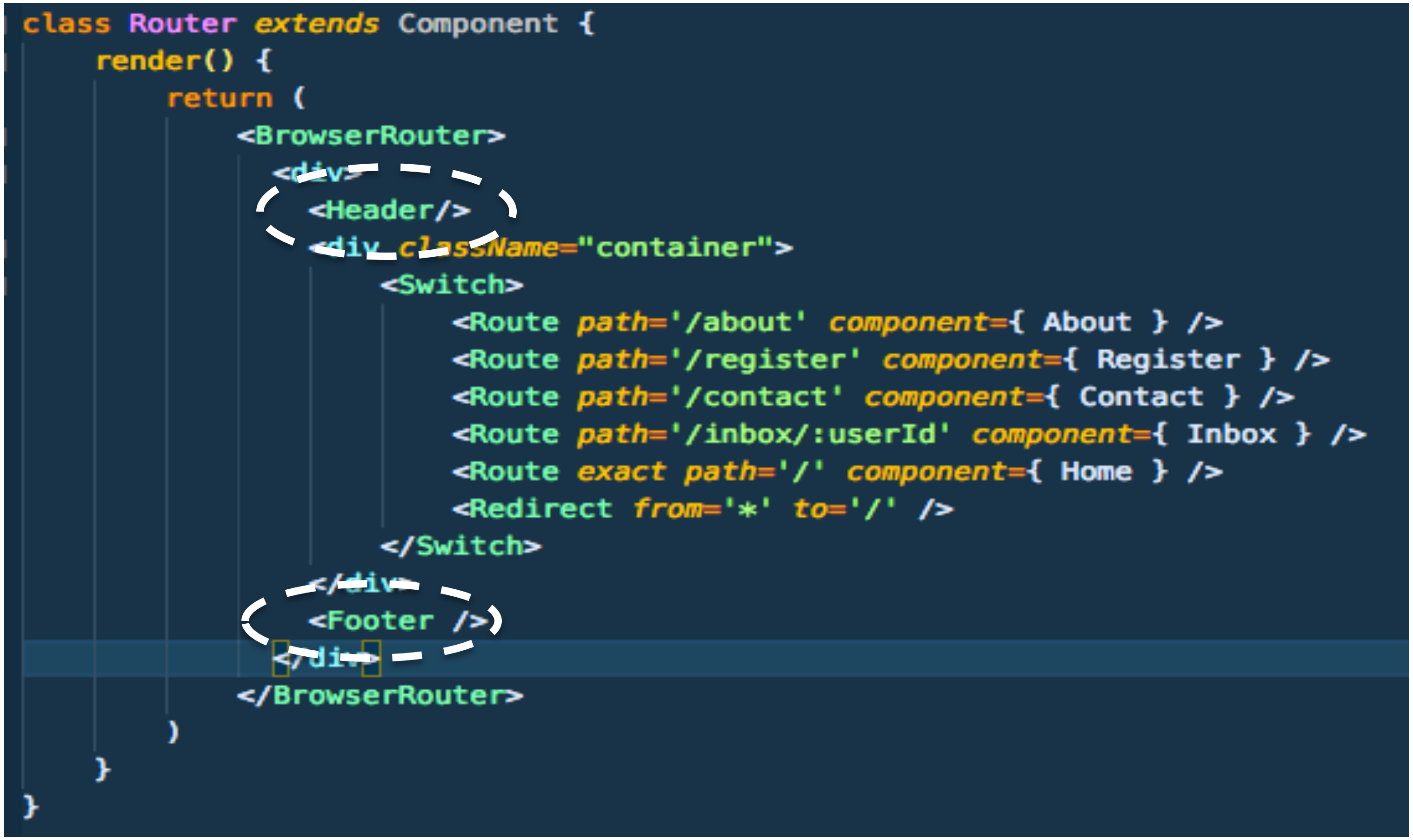
Inbox page

Messages for user: 1234

The ReactJS libraryby Facebook Contact Us Nav X Nav Y News

Persistent elements/components

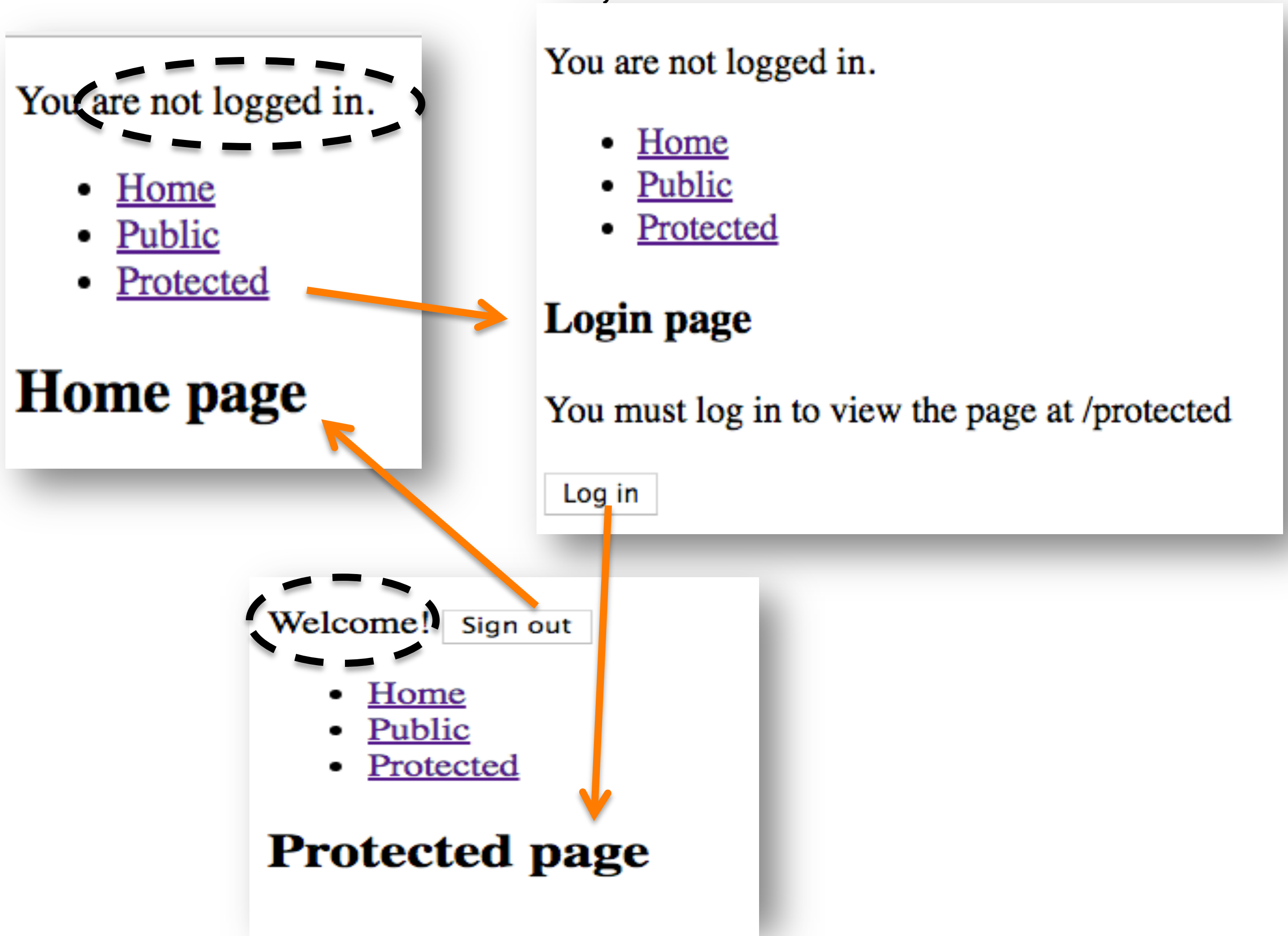
- **Ref.** src/sample8



```
class Router extends Component {
    render() {
        return (
            <BrowserRouter>
              <div>
                <Header/>
                <div className="container">
                    <Switch>
                        <Route path='/about' component={ About } />
                        <Route path='/register' component={ Register } />
                        <Route path='/contact' component={ Contact } />
                        <Route path='/inbox/:userId' component={ Inbox } />
                        <Route exact path='/' component={ Home } />
                        <Redirect from='*' to='/' />
                    </Switch>
                </div>
                <Footer />
              </div>
            </BrowserRouter>
        )
    }
}
```

Protected Routes

- **Not native to React Router; A custom solution.**



You are not logged in.

- Home
- Public
- Protected

Home page

You are not logged in.

- Home
- Public
- Protected

Login page

You must log in to view the page at /protected

Welcome!

- Home
- Public
- Protected

Protected page

Protected Routes

- **See** /src/sample9/
  - **v1 – Skeleton.**
  - **v2 - basic <PrivateRoute> component, Redirects to skeleton Login page**
  - **v3 - Login supported but redirects to / (root) always; Protected page now accessible.**
  - **v4 - Login now redirects to selected protected page.**
  - **v5 – Signout supported.**

- **To step through sample versions:**

```
$ git checkout sample9-vX e.g. git checkout sample9-v3
```

- **To return to normal:**

```
$ git checkout master
```

Summary

- **React Router (version 4) adheres to React principles:**
  - **Declarative UI**
  - **Component composition**
  - **The event → state change → re-render**
- **Main components - <Router>, <Route>, <Link>**
- **Additional props: this.props.match.params; this.props.history, this.props.location**